CS 596: Theoretical Machine Learning

Lecturer: Abhishek Bhrushundi
Scribe: Pranjal Awasthi

Lecture # 1 & 2
September , 2016

1 Introduction to Probably Approximately Correct (PAC) Learning

The PAC model introduced by Valiant [1] is the standard framework to study classification problems in machine learning. We are given an instance space $\mathcal{X}$ that could be finite or infinite. An example of a finite $\mathcal{X}$ is the Boolean hypercube, i.e., the set $\{0,1\}^n$. The Euclidean space $\Re^d$ is an example of an infinite instance space. Our goal is to approximately learn a function $h^* : \mathcal{X} \mapsto \{0,1\}$. We don't have direct access to h^*. However, we do have a training set S that is labeled according to the target function. Furthermore, we also know that h^* belongs to class of functions H.

Specifically, we have $S = \{(x_1, y_1), (x_2, y_2), \ldots (x_m, y_m)\}$, where $x_i \in \mathcal{X}$ and $y_i = h^*(x_i)$. We want to design a learning algorithm that takes as input the set S and outputs a function $h : \mathcal{X} \mapsto \{0,1\}$. The hope is that given enough data h will help us classify future examples as well as h^*. In order to make this more precise, we need to make sure that the future examples are "similar" to the training set. Indeed, if the future data was completely unrelated to the training set, then no learning algorithm can do well.

A neat way to enforce similarity of training and test data is via a probability distribution. We will assume that there is an unknown distribution D over $\mathcal{X}$ and each example x_i is an independent sample drawn according to D. Hence, our training set consists of m independent and identically distributed random variables, i.e. the x_i's, that are labeled according to h^*. To measure the performance of any function h that is output by the learning algorithm, we define the error of h as the probability that h disagrees with h^* on a new example drawn from the same distribution D. More formally we have $err(h) = Pr_{x\sim D}[h(x) \neq h^*(x)]$. Hence, our goal is to design an algorithm that can process the training set S and with high probability, output a hypothesis h of low error.

Definition 1 (PAC Learning)**.** *An algorithm A PAC-learns a class of functions H if: for any $\epsilon > 0, \delta > 0$, any target $h^* \in C$, any distribution D over $\mathcal{X}$, on input a training set S, A produces, with probability $\geq 1 - \delta$, a hypothesis h such that $err(h) \leq \epsilon$.*

We say that A efficiently PAC-learns C if it runs in time polynomial in $|S|$ and uses a small amount of training data, i.e., $|S|$ is polynomial in $\frac{1}{\epsilon}, \frac{1}{\delta}$, and the complexity of H[1]. We say that A is a proper learner if the output function h always lies in H. We will later see instances where it is helpful, for computational reasons, to output a function in a larger class.

2 Empirical Risk Minimization (ERM) as a Universal Learning Algorithm

Given a finite training set S, a natural approach to learning is to find the function in H that has the minimum error on S. One would then hope that this function will also do well on

[1] This will become more clear later on.

future data from the same distribution. This general principle is known as *Empiricial Risk Minimization*. For a given $h \in H$, define its empirical error as $err_S(h) = \frac{1}{|S|}\sum_{i=1}^{|S|} I(h(x_i) \neq y_i)$. Here $I()$ is the indicator function that takes value 1 when $h(x_i) \neq y_i$ and 0 otherwise. So, $err_S(h)$ measures the fraction of errors that the function h makes on the training data S. The ERM principle outputs h that minimizes this error, i.e., $h = argmin_{f \in H} err_S(f)$.

Theorem 1. *Given a finite function class H, ERM PAC-learns H provided the size of the training set $m \geq \frac{1}{\epsilon}\log(\frac{|H|}{\delta})$.*

Proof. Notice that given a training set S, ERM will always find an h such that $err_S(h) = 0$. This is because we know that $h^* \in H$ and achieves zero error on every example. This is often known as the realizable case. Later we will remove this assumption. Let $BAD \subset H$ be the hypotheses with true error more than ϵ. That is, $BAD = \{h \in H : err(h) > \epsilon\}$. We want to bound $Pr_S[A(S) \in BAD]$, where $A(S)$ is the hypothesis output by ERM. By union bound, we can say that $Pr_S[A(S) \in BAD] \leq \sum_{h \in BAD} Pr_S[A(S) = h]$. Now let's bound $Pr_S[A(S) = h]$. For A to output h, it must achieve 0 error on m randomly drawn i.i.d. samples from D. Since, h as at least an ϵ chance of making a mistake on every example, the probability that it gets m i.i.d. examples correctly is at most $(1-\epsilon)^m \leq e^{-\epsilon m}$. Hence, we get that $Pr_S[A(S) \in BAD] \leq \sum_{h \in BAD} e^{-\epsilon m} \leq |BAD|e^{-\epsilon m} \leq |H|e^{-\epsilon m}$. Setting this to be less than δ, we get that ERM will succeed in finding a good hypothesis provided $m \geq \frac{1}{\epsilon}\log(\frac{|H|}{\delta})$. □

2.1 Why not minimize error over all functions?

A natural question to ask is what happens if one outputs the hypothesis that minimizes $err_S(h)$ over the space of all functions and not just H. This might lead to a phenomenon known as "overfitting" wherein the error on the training set is a high optimistic estimate of the true error. As a simple example, consider a universal learning algorithm that takes as input a training set S and outputs a hypothesis h that does the following: Given a new example x, if $x \in S$, output the label of x, otherwise output a random 0/1 label. Clearly, this learner has zero error on every training set. However, the true error of this algorithm will be $\frac{1}{2}$ on any non-trivial problem.

What goes wrong in the proof above? Since we are not minimizing error over just H anymore, we need to take union bound over all possible hypotheses that A might ouptut. This will be an infinite set and hence the bound on m above does not hold.

3 Additional Readings

- Chapters 2 and 3 from the book by Shai Shalev-Shwartz and Shai Ben-David. See online copy here: http://www.cs.huji.ac.il/ shais/UnderstandingMachineLearning/understanding-machine-learning-theory-algorithms.pdf
- Chapters 1 and 3 from the book by Micheal Kearns and Umesh Vazirani. https://www.amazon.com/In Computational-Learning-Theory-Press/dp/0262111934

References

[1] L. G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.